Appendix 1 : code developed in python to solve ODE system for biofilm modeling

```
import scipy
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from lmfit import minimize, Parameters, Parameter, report_fit
import pandas as pd


def f(y, t, paras):
"""
Definition of states variables
"""
X = y[0] #biomass (g/L)
S = y[1] #residual glucose (g/L)
Iso = y[2] #isopropanol (g/L)
B = y[3] #butanol concentration (g/L)
AA= y[4] # acetate concentration (g/L)
AB = y[5] # Butyrate concentration (g/L)
E = y[6] # Ethanol concentration (g/L)

try:
mumax = paras['mumax'].value
Yxs = paras['Yxs'].value
Ybuts = paras['Ybuts'].value
Yes = paras['Yes'].value
Yis = paras['Yis'].value
ks = paras['ks'].value
Bmax = paras['Bmax'].value

kAB = paras['kAB'].value
kAA = paras['kAA'].value
a = paras['a'].value
b = paras['b'].value

Yab_but = paras['Yab_but'].value
Yi_aa = paras['Yi_aa'].value


except KeyError:
mumax,Yxs,Ybuts,Yes,Yis,ks,Bmax,kAB,kAA,a,b,Yab_but= paras

"""
kinetic rates
"""

μ = mumax*(1-(B/Bmax))*(S/(ks+S))

rbut = μ*(Ybuts/Yxs)
```

```
ri = μ*(Yis/Yxs)

rbs = (μ)*X*a-rbut*(AB/(AB+kAB))*(S/(ks+S))*X*(1/Yab_but)

raa = (μ)*X*b-ri*(AA/(AA+kAA))*(S/(ks+S))*X*(1/Yi_aa)

re = μ*X*(Yes/Yxs)

"""
Mass balance analysis
"""

dX_dt = μ*X
dS_dt = (-1/Yxs)*(μ)*X
dB_dt = rbut*X #+ (AB/(AB+kAB))*X
dI_dt = ri*X #+ (AA/(AA+kAA))*X
dAA_dt = raa
dAB_dt = rbs
dE_dt = re

F= np.array([dX_dt,dS_dt,dI_dt,dB_dt,dAA_dt,dAB_dt,dE_dt])
return F


def g(t, x0, paras):
"""
Solution to the ODE x'(t) = f(t,x,k) with initial condition x(0) =
x0
"""
x = odeint(f, x0, t, args=(paras,))
return x


def residual(paras, t, data):

"""
compute the residual between actual data and fitted data
"""

x0 = paras['X0'].value,
paras['S0'].value,paras['Iso_0'].value,paras['But_0'].value,paras['A
A_0'].value,paras['AB_0'].value, paras['E_0'].value
model = g(t, x0, paras)

# you only have data for one of your variables

return (model - data).ravel()


"""
Initial conditions
"""
```

```
X0 = 0.3
S0 = 60
Iso_0 = 0.01
But_0 = 0.01
AB_0 = 0.01
AA_0 = 2.1
E_0 = 0.01
x0 = [X0,S0,Iso_0,But_0,AA_0,AB_0,E_0]
ts = np.linspace(0, 75, 1000)

"""
Import of experimental data
"""
#Lecture du fichier csv
data_1 = pd.read_csv("data_4.csv",sep=';')#,index_col = 0, header=
0)
data_2 = pd.DataFrame()
data_2=data_1[['MS (g/L)','Glucose i','Isopropanol (g/L)','Butanol
(g/L)','Acetate (g/L)','Butyrate (g/L)','Ethanol (g/L)']]

#Transformation du data frame en matrice
data_exp= data_2.to_numpy()
t_exp = data_1.loc[:,'Temps (h)']
t_exp = t_exp.values


"""
Parameters definition
"""

params = Parameters()

params.add('X0', value=X0, vary=False)
params.add('S0', value=S0, vary=False)
params.add('Iso_0', value=Iso_0, vary=False)
params.add('But_0', value=But_0, vary=False)
params.add('AA_0', value=AA_0, vary=False)
params.add('AB_0', value=AB_0, vary=False)
params.add('E_0', value=E_0, vary=False)

params.add('mumax',value = 0.15, vary = True)

params.add('Yxs', value = 0.15, vary = True)
params.add('Ybuts',value = 0.17, vary = True)
params.add('Yes',value = 0.003,min = 0, max = 0.05, vary = False)
params.add('Yis',value = 0.11, min = 0.06,max=0.2, vary = True)
params.add('ks',value = 2,min =1,max =10, vary = False)
params.add('Bmax', value = 7.2, vary= True)

params.add('kAB',value = 0.3, min = 0.05, max = 5,vary = True)
params.add('kAA',value = 0.2, min = 0.01, max = 5, vary = True)
```

```
params.add('a',value = 1, min = 0.01, max = 30, vary = False)
params.add('b',value = 0.5,  max = 30, vary = False)

params.add('Yab_but',value = 0.7,  max = 0.85, min = 0, vary =
False)
params.add('Yi_aa',value = 1,  max = 0.85, min = 0, vary = False)


"""
Results of data fiting
"""

result = minimize(residual, params, args=(t_exp,data_exp),
method='Least_square')  # leastsq minimization
paras = result.params #Set of parameters obtained after regression

data_fitted = odeint(f,x0,ts,args=(paras,)) # check results of the
fit
IBEsim = data_fitted[:, 6] + data_fitted[:, 3] + data_fitted[:, 2]
IBEexp = data_exp[:, 6] + data_exp[:, 3]+ data_exp[:, 2]

"""
Plots of experimental and simulated data
"""

# plot of biomass and glucose concentration
plt.figure(1)
fig, ax1 = plt.subplots()
plt.grid()
plt.xticks(size =14)
plt.yticks(size =14)

ax2 = ax1.twinx()
ax1.plot(ts, data_fitted[:, 0], '-', linewidth=1.3, color='black',
label=' biomass (mod)')
ax1.scatter(t_exp,data_exp[:,0],s = 60,color =
'white',marker='d',edgecolor='black', label = 'biomass(exp)')
ax2.plot(ts, data_fitted[:, 1], '--', linewidth=1.3, color='black',
label='glucose (mod)')
ax2.scatter(t_exp,data_exp[:,1],s = 45,color =
'white',marker='s',edgecolor='black',label = 'glucose (exp)')
ax1.set_xlabel('time (h)', size = 14)
ax1.set_ylabel('Biomasse concentration (g/L)', color='black',size =
14)
plt.yticks(size =14)

ax2.set_ylabel('residual glucose concentration (g/L)',
color='black', size =14)
plt.yticks(size =14)
plt.show()
```

```
# plot of acetate and butyrate concentration over time
plt.figure()
plt.grid()
plt.plot(ts, data_fitted[:, 4], '-', linewidth=1, color='black',
label='acetate (mod)')
plt.scatter(t_exp,data_exp[:,4],s =45,color =
'white',marker='s',edgecolor='black', label = 'acetate (exp)')
plt.plot(ts, data_fitted[:, 5], '--', linewidth=1, color='black',
label='butyrate(mod)')
plt.scatter(t_exp,data_exp[:,5],s =60,color = 'white', marker
='d',edgecolor='black', label = 'butyrate (exp)')
plt.xlabel("time (h)", size = 14)
plt.ylabel('Acetate, Butyrate (g/L) ', size = 14)
plt.xticks(size =14)
plt.yticks(size =14)
plt.legend(bbox_to_anchor=(1, -0.15),loc='best',frameon= False,ncol
=2,prop={'size': 14})
plt.show()

# plot of isopropanol, butanol and ethanol concentration over time
plt.figure(3)
plt.grid()
plt.plot(ts, data_fitted[:, 2], '-',  color='black',
label='isopropanol (mod)')
plt.scatter(t_exp,data_exp[:,2],color = 'white',s =60,marker='*',
edgecolor = 'black',label = 'isopropanol (exp)')
plt.plot(ts, data_fitted[:, 3], '--', linewidth=1, color='black',
label='butanol (mod)')
plt.scatter(t_exp,data_exp[:,3],color = 'white',marker =
'o',edgecolor='black', label = 'butanol (exp)')
plt.plot(ts, data_fitted[:, 6], '-', linewidth=1, color='grey',
label='ethanol (mod)')
plt.scatter(t_exp,data_exp[:,6],color = 'grey',marker ='+', label =
'ethanol (exp)')
plt.plot(ts, IBEsim, '-', linewidth=1, color='black', label='Total
solvent (mod)')
plt.scatter(t_exp,IBEexp,color = 'white', marker
='D',edgecolor='black', label = 'total solvent (exp)')
plt.xlabel("time (h)", size = 14)
plt.xticks(size =14)
plt.ylabel('Butanol, isopropanol, ethanol (g/L)', size = 14)
plt.yticks(size =14)
plt.legend(bbox_to_anchor=(1.3, -0.15),loc='best',frameon=
False,ncol =3,prop={'size': 14})
plt.show()


"""
Analysis of fit quality
"""
```

```
report_fit(result) # parameters of confidence intervals  and fit
statistics
res = result.residual # analysis of parameters residuals

nres = np.linspace(0,len(res),len(res)) #plot of residuals
plt.scatter(nres,res,marker ='+',edgecolor='black')
plt.xlabel("n point")
plt.ylabel('résidu du modèle')

print("la moyenne des résidus est de : ",np.mean(res))
norm = scipy.stats.shapiro(res)
print("la statisqtique de test p.value est égale à",norm)
```